Contents lists available at ScienceDirect

Journal of Power Sources

journal homepage: www.elsevier.com/locate/jpowsour

Extended operating range for reduced-order model of lithium-ion cells

James L. Lee, Lukas L. Aldrich, Kirk D. Stetzel, Gregory L. Plett*

Department of Electrical and Computer Engineering, University of Colorado Colorado Springs, Colorado Springs, CO 80918, United States

HIGHLIGHTS

- Battery-management systems require accurate cell models over a wide operating range.
- A previously reported physics-based model is accurate over only a narrow range.
- We use model blending to extend the operating window of the physics-based model.
- We show that the blended model is stable and accurate over a broad operating range.

ARTICLE INFO

Article history:
Received 10 August 2013
Received in revised form
27 December 2013
Accepted 31 December 2013
Available online 9 January 2014

Keywords:
Lithium ion battery
Electrochemical model
Battery management systems
Model blending
Transcendental transfer function
Model order reduction

ABSTRACT

In a previous paper, we developed a method to produce a physics-based one-dimensional discrete-time state-space reduced-order model (ROM) of a lithium-ion cell. The method relies on linearizing the standard porous-electrode equations around a fixed state-of-charge (SOC) and operating temperature setpoint. The ROM is able to track a highly dynamic input accurately near the linearization setpoint, but its performance degrades as either the cell's SOC or temperature move away from this linearization point.

This paper describes a way to extend the accuracy of the ROM over a wide range of SOCs and temperatures using a model-blending approach. Our results demonstrate that the approach accurately models the cell's voltage and internal electrochemical variables over a wide range of temperature and SOC, with little added computational complexity.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

Battery chargers and battery-management systems rely on mathematical models of their battery cells to be able to determine dynamic operational limits within which the battery pack may be safely operated. Current battery-management systems use equivalent circuit models [1]. The advantage of these models is that they are simple enough computationally to be incorporated into real-time control systems, and thus can be used in methods to estimate many cell properties such as state-of-charge (SOC) and state-of-health, as well as voltage-based power limits (see, for example, [2–4]).

Equivalent-circuit models can produce accurate predictions of cell voltage; however, they do not provide insight into the internal electrochemical variables of the cell. On the other hand, porous-electrode models do have this ability. Doyle, Fuller, and Newman [5,6] have developed such a physics-based porous-electrode model, which comprises coupled nonlinear partial-differential equations (PDEs). This model describes the electrochemical internal dynamics of the cell in addition to being able to predict cell voltage.

Control algorithms based on knowledge of the internal electrochemical state have the potential to expand the performance and extend the life of cells. They can predict power limits with respect to electrode surface depletion/saturation conditions and with respect to side reactions responsible for damage and sudden loss of power [7,8]. For example, it is shown in Ref. [9] that electrochemically limited pulse charging a 6 Ah cell to the same negative-electrode phase-potential $\phi_s - \phi_e$ at the negative-electrode/separator boundary as encountered at equilibrium at 100% SOC increases usable charge power by 22% and usable energy by 212% versus voltage-limited charging.

* Corresponding author. Tel.: +1 719 255 3468; fax: +1 719 255 3589.
E-mail addresses: jlee3@uccs.edu (J.L. Lee), laldrich@uccs.edu (L.L. Aldrich), kstetzel@uccs.edu (K.D. Stetzel), gplett@uccs.edu (G.L. Plett).

0378-7753/$ – see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jpowsour.2013.12.134

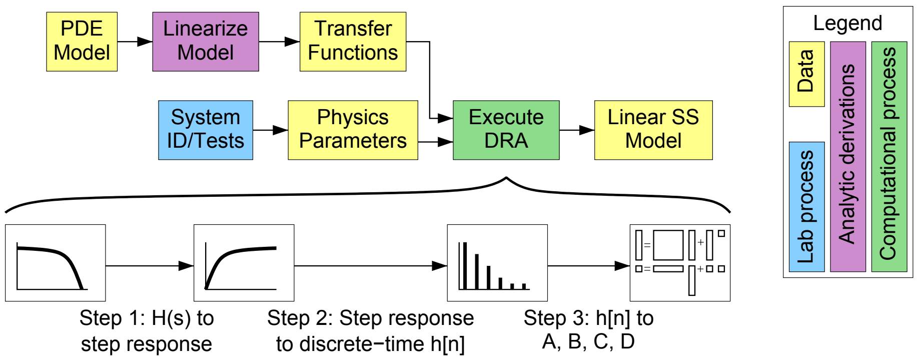


Fig. 1. Approach to generating the linear state-space model.

However, the computational complexity of the porous-electrode PDE models precludes their use by real-time control systems. Instead, accurate reduced-order approximate models, which can predict both the cell voltage and internal electrochemical variables, are needed. Over a series of papers, we are presenting a methodology to develop such models.

The first paper in the series introduced the "discrete-time realization algorithm" (DRA) as a method for converting a transcendental transfer function into a discrete-time state-space reduced-order model [10]. It is further shown in Ref. [10] that the resulting model is a globally optimum p-rank approximation to the transfer function it approximates, by the Schmidt–Mirsky theorem [11]. The second paper showed how to find transcendental transfer functions corresponding to lithium-ion internal cell dynamics [12]. The scope of the work in the second paper was limited to showing how to create an ROM that is linearized to give accurate predictions around a single SOC and temperature set-point only. This ROM was able to predict cell voltage as well as cell potentials $\phi_s(x,t)$ and $\phi_e(x,t)$, lithium concentrations $c_{s,e}(x,t)$ and $c_e(x,t)$, and intercalation flux $j(x,t)$ for any desired combination of one-dimensional internal cell locations x. This was accomplished using a fifth-order linear discrete-time state-space model, plus some nonlinear algebraic corrections. The resulting reduced-order model (ROM) is simple enough computationally for use in real-time estimation of the electrochemical variables. Our results demonstrated that the ROM performs well if the temperature remains constant and the SOC does not vary significantly from the linearization point.

This paper is the third in the series of planned works. Here, we present a method to extend the accuracy of the model over a wide range of both SOC and operating temperatures. We accomplish this by generating individual models at specific states-of-charge and temperatures and then blend the models using the real-time estimates of the SOC and temperature. A planned fourth paper will show how to incorporate heat generation and heat flow into the model.

The paper is organized as follows. In Section 2, we review the approach used to produce the discrete-time reduced-order model at a particular operational setpoint, and how to apply this model to an arbitrary current input. In Section 3, we discuss the impact of changing temperature and SOC on each ROM. In Section 4, we describe the specific implementation of the proposed model-blending approach, and demonstrate that this approach produces a stable system. Results using three different current and temperature scenarios are presented in Section 5.

2. Background: the single setpoint ROM

An overview of the steps required to generate a physics-based reduced-order linear state-space model at a specific operating setpoint is illustrated in Fig. 1. The first step is to determine the physics parameters of the cell, which include factors such as electrode and particle dimensions, conductivities, diffusivities, porosities, open-circuit-potential relationships, and so forth. These parameters are found either by direct measurements and electrochemical experiments or, in the cases where that is not feasible, by running tests on the cell and using system identification techniques to estimate the parameter values. The values of these physical parameters are input to the PDE porous-electrode model equations, which are represented as "PDE Model" in the figure. These equations describe the electrochemical dynamics of the cell and allow us to solve for five electrochemical variables within the cell: the reaction flux $j(x,t)$, solid and electrolyte lithium concentration $c_{s,e}(x,t)$ and $c_e(x,t)$, and solid and electrolyte potentials $\phi_s(x,t)$ and $\phi_e(x,t)$. These nonlinear coupled PDEs can be solved using finite-element software, for example, but this is computationally too complex for use in a real-time controller. Instead, we must find a reduced-order model.

In the next step, we follow the approach introduced in Refs. [7,13] and convert the PDE model into transcendental transfer functions after making two simplifying assumptions: 1) that the potential in the electrolyte is not dependent on the concentration of lithium in the electrolyte, and 2) that the nonlinear equations can be linearized. We then derive closed-form Laplace-domain transcendental transfer functions from these linearized equations. The discrete-time realization algorithm (DRA), based on the Ho–Kalman algorithm [14], is used to convert these transcendental transfer functions into an optimal discrete-time state-space realization. A detailed description of the DRA is given in Ref. [10]. A brief summary of the steps is as follows:

1. Sample the continuous-time transfer function in the frequency domain at a high rate, and take the inverse discrete Fourier transform (IDFT) to get an approximation to the continuous-time impulse response. Then, form the continuous-time step response from the continuous-time impulse response.
2. Compute the discrete-time pulse response values from the continuous-time step response, assuming a sample and hold circuit connected to the system input.
3. Generate a discrete-time state-space realization using the deterministic Ho–Kalman algorithm. This algorithm returns the reduced-order **A**, **B**, and **C** matrices from the discrete-time pulse-response sequence in Step 2. The order of the system is determined from the sorted singular values of the Hankel matrix that is computed as part of the algorithm. The **D** matrix is found by the initial value theorem.

The resulting ROM comprises a linear dynamic part and some algebraic nonlinear corrections. The optimal reduced-order discrete-time linear state-space model has the form

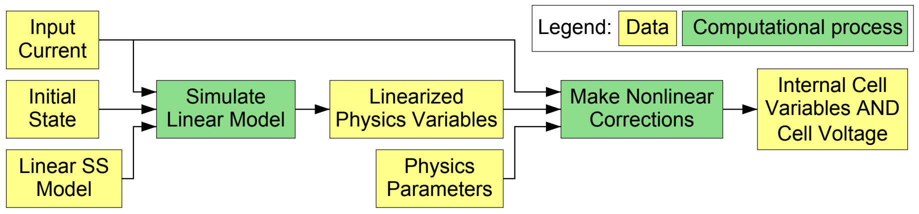


Fig. 2. Simulating a reduced-order-model.

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \quad (1)$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k, \quad (2)$$

where $\mathbf{u}_k$ is the model "input vector" at the kth sample period (which includes the applied cell current i_{app} as one component, but which may also include other inputs if desired), $\mathbf{x}_k$ is the "state vector" of the model at discrete-time index k, and $\mathbf{y}_k$ is the "output vector" of the model at discrete-time index k. Equation (1) is called the "state equation" of the model, and Equation (2) is called the "output equation" of the model. The state equation describes the dynamics of the model, which are summarized by the state $\mathbf{x}_k$. The output equation computes linearized variables of interest from the model's state at any point in time.

One element of the state vector $\mathbf{x}_k$ is an integrator, which physically models the lithium concentration in the negative electrode, and hence also cell state-of-charge. The other state-vector elements are abstracted quantities with no direct physical interpretation of which we are aware. However, the output vector $\mathbf{y}_k$ contains signals that are linearized versions of the internal cell variables of interest, such as reaction flux $j(x,t)$, solid and electrolyte lithium concentration $c_{s,e}(x,t)$ and $c_e(x,t)$, and solid and electrolyte potentials $\phi_s(x,t)$ and $\phi_e(x,t)$ at whatever cell locations x are required in an application.

Thus, Fig. 2 depicts the process followed to estimate the cell voltage and the five electrochemical variables at any desired location throughout the cell for a given input-current profile. The linear state-space model, along with the input current $\mathbf{u}_k$, and the initial state $\mathbf{x}_0$ produce the linearized physical variables $\mathbf{y}_k$. Using the physics parameters, we make nonlinear corrections as described in Ref. [12] to obtain an estimate of the true internal electrochemical variables.

Simulation results demonstrate that ROM cell voltage predictions and the ROM internal electrochemical variable predictions using a fifth-order model match very closely with results obtained by simulating the full nonlinear porous-electrode partial differential equations. In Ref. [12], we demonstrated that a fifth-order ROM could accurately track the full order model of a highly dynamic driving profile with cell-voltage root-mean-squared (RMS) error of 1.4 mV. We also showed close tracking of the internal cell variables.

3. Impact of temperature and SOC change on the ROM

The ROM described in Section 2 was generated by linearizing the porous-electrode model equations at a specific cell SOC and temperature. When used to predict cell variables in conditions close to the SOC–temperature setpoint, the ROM works very well. However, its predictions become less accurate when the SOC and/or temperature differ significantly from the linearization setpoint. For some applications, this may not be important—cell SOC and temperature may experience only minor fluctuation. However, others such as hybrid-electric-vehicle battery packs may experience large temperature variation, even though the SOC variation may be small. Still other applications such as electric-vehicle battery packs will experience large variation in both temperature and SOC. Therefore, there is a need to have models that work well over large variations in temperature and/or SOC. It is the purpose of this paper to describe a simple approach that combines information from multiple ROMs to deliver accurate predictions over a wide SOC and temperature operation range.

Table 1
Activation energies for the temperature-dependent electrochemical parameters, in J mol^{-1}.

	Negative electrode	Separator	Positive electrode
E_{act,D_e}		10,000	
$E_{\text{act},\kappa}$		20,000	
E_{act,D_s}	4000	–	20,000
$E_{\text{act},k}$	30,000	–	30,000

We first consider how to model temperature-dependent parameter values.[1] In Botte et al. [15], the authors identified the temperature-dependent cell parameters to be the solid and electrolyte diffusivity (D_s and D_e), the electrolyte conductivity (κ) and the reaction rate constant (k). In Ref. [16] the temperature-dependent parameter change is modeled using the empirical Arrhenius equation,

$$\Phi = \Phi_{\text{ref}} \exp\left[\frac{E_{\text{act},\Phi}}{R}\left(\frac{1}{T_{\text{ref}}} - \frac{1}{T}\right)\right]. \quad (3)$$

In this equation, Φ represents any of the temperature-dependent parameters and $E_{\text{act},\Phi}$ is the corresponding activation energy, which can be thought of as a measure of the sensitivity of the parameter to a temperature change. Φ_{ref} is the value of the parameter at the reference temperature, T_{ref}, which is typically 298.15 K. Table 1 shows the activation energies estimated by Gu and Wang in Ref. [16] for the particular cell used by Botte et al. in Ref. [15].

We next model cell state-of-charge, which is defined, based on negative-electrode quantities, as

$$\text{SOC} = \frac{\frac{c_{s,\text{avg}}(t)}{c_{s,\max}} - \theta_{\min}}{\theta_{\max} - \theta_{\min}}, \quad (4)$$

where $\theta_{\min}$ and $\theta_{\max}$ are the stoichiometric limits of the negative electrode in the fully discharged and fully charged states, respectively, $c_{s,\max}$ is the maximum concentration of lithium in the solid particles of the negative electrode, and $c_{s,\text{avg}}(t)$ is the average concentration of lithium in the solid particles of the negative electrode

[1] Note that it is not the purpose of this paper to present a fully coupled thermal–electrochemical reduced-order model of a lithium ion cell, where the model predicts the heating and cooling of the cell via internal heat generation and external heat transfer. That is the topic of a planned future paper. This paper uses measured temperature as an input to the model, and bases its predictions on that temperature value.

at time t. The value of $c_{s,\max}$ is a physical parameter of the cell; the values of $\theta_{\min}$ and $\theta_{\max}$ are design variables for the "operating window" of electrode usage and are chosen as a tradeoff between maximizing capacity and minimizing degradation rates; the value of $c_{s,\mathrm{avg}}(t)$ is an internal state calculated at each time step. Variations in state-of-charge have biggest impact on the dynamics of variables that depend heavily on the open-circuit-potential of the electrode. However, some cell parameters, such as solid diffusivity D_s may also exhibit state-of-charge dependence that must be captured by the model for best predictions.

4. Model blending

In order to model the cell over a wide range of both temperature and SOC, we use a model-blending approach. The basic idea is to precompute ROMs at multiple SOC and temperature setpoints and then—during operation—use these to generate a best "average" ROM specialized for the present instantaneous SOC and temperature. Note that the computational requirements of creating an ROM from a specific SOC and temperature setpoint is high; the computation requirement of averaging precomputed ROMs is low—this approach avoids computing the ROM in real-time during operation.

Although it is considered an ad-hoc method, model-blending is one of the most popular approaches to modeling nonlinear systems [17,18]. This method typically works well when the scheduling variables change slowly during operation as is the case with cell SOC and temperature, and when the variation in model parameter values is smooth with respect to a change in the scheduling variables, which we will also see to be true of our application.

4.1. Blending the models

Individual reduced-order models are generated *a priori* using the DRA over the expected operating range of temperatures and states of charge. For simplicity, we assume that the setpoints are generated as the Cartesian product of a temperature vector and an SOC vector, so that they fall on a rectangular grid.[2] For each individual reduced-order model, the temperature-dependent cell parameters are determined from their reference values by the Arrhenius equation; SOC-dependent parameters are specified from the setpoint SOC value.

These precomputed models are blended in real time using bilinear interpolation to generate a time-varying state-space model, as illustrated in Fig. 3. We define SOC_0 to be the nearest SOC setpoint value among the precomputed models that is less than or equal to the cell's present operating SOC value, and SOC_1 to be the nearest SOC setpoint that exceeds the cell's present operating SOC value. Similarly, we define T_0 to be the nearest temperature setpoint value among the precomputed models that is less than or equal to the cell's present operating temperature, and T_1 to be the nearest temperature setpoint value that exceeds the cell's present operating temperature. We then define blending factors x and y as

$$x = \frac{\mathrm{SOC}-\mathrm{SOC}_0}{\mathrm{SOC}_1-\mathrm{SOC}_0} \quad \text{and} \quad y = \frac{T-T_0}{T_1-T_0}.$$

The value of the time-varying blended $\mathbf{A}_k$ matrix is found from

$$\mathbf{A}_k = (1-y)\big((1-x)\mathbf{A}_{0,0}+x\mathbf{A}_{1,0}\big)+y\big((1-x)\mathbf{A}_{0,1}+x\mathbf{A}_{1,1}\big),$$

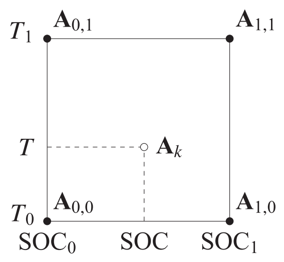


Fig. 3. Bilinear transformation of the **A** matrix.

where $\mathbf{A}_{0,0}$ is the **A** matrix of the precomputed model at SOC_0 and T_0, $\mathbf{A}_{0,1}$ at SOC_0 and T_1, $\mathbf{A}_{1,0}$ at SOC_1 and T_0, and $\mathbf{A}_{1,1}$ at SOC_1 and T_1. The time-varying blended $\mathbf{B}_k$, $\mathbf{C}_k$, and $\mathbf{D}_k$ matrices can be found in the same manner (although we see some simplifications in the next section, which make it unnecessary to blend $\mathbf{B}_k$). Equations (1) and (2) are then modified with these time-varying blended matrices to become

$$\mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k+\mathbf{B}_k\mathbf{u}_k \tag{5}$$

$$\mathbf{y}_k = \mathbf{C}_k\mathbf{x}_k+\mathbf{D}_k\mathbf{u}_k. \tag{6}$$

Fig. 4 illustrates the real-time aspect of the overall model-blending approach. During operation, the present cell SOC and temperature are used to generate blended $\mathbf{A}_k$, $\mathbf{B}_k$, $\mathbf{C}_k$ and $\mathbf{D}_k$. The model state vector $\mathbf{x}_k$ is updated using these time-varying matrices, and the linearized outputs $\mathbf{y}_k$ are computed from the updated state vector. The SOC calculated at each time step from the internal states is fed back into the linear model as an input to this blending process.

4.2. Sorting of the model

One complication when implementing the model-blending scheme arises from the fact that state-space models are not unique. An infinite number of different state descriptions with corresponding $\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}\}$ represent the same input–output relationship between $\mathbf{u}_k$ and $\mathbf{y}_k$. This poses a potential problem when model blending because for the method to work all elements of the matrix $\mathbf{A}_{0,0}$ must be consistent in meaning with the corresponding elements of the matrices $\mathbf{A}_{1,0}$, $\mathbf{A}_{0,1}$, and $\mathbf{A}_{1,1}$. If not, unrelated elements will be averaged together, producing a meaningless result. The DRA algorithm itself does not guarantee that models generated at different temperature and SOC setpoints will exhibit a consistent state-space description.

There is a simple remedy, however, which is to transform all precomputed models into a common framework. We do so as follows. We begin by supposing that a linear discrete-time state-space model produced by the DRA is of the form

$$\mathbf{x}_{k+1}^{(0)} = \mathbf{A}^{(0)}\mathbf{x}_k^{(0)}+\mathbf{B}^{(0)}\mathbf{u}_k$$

$$\mathbf{y}_k = \mathbf{C}^{(0)}\mathbf{x}_k^{(0)}+\mathbf{D}\mathbf{u}_k.$$

The superscript "(0)" on several of the model terms indicates that these matrices and signals arise from the untransformed model produced directly from the DRA. We will use superscripts "(1)," "(2)," and "(3)" in the following to indicate different stages of model transformation.

For the first transformation, we define a new state vector $\mathbf{x}_k^{(1)}$, such that $\mathbf{x}_k^{(0)} = \mathbf{T}^{(1)}\mathbf{x}_k^{(1)}$, where $\mathbf{T}^{(1)}$ is some square invertible

[2] This is not a strict requirement, but it makes the blending straightforward as the standard bilinear interpolation method may be used. Interpolation is more complicated if a non-rectangular grid of points is used, but can still be done.

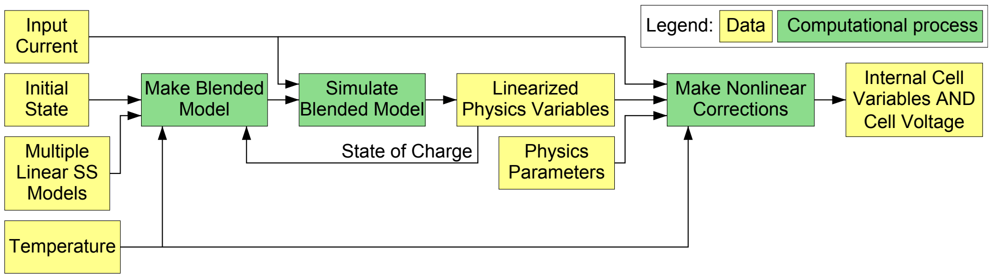


Fig. 4. Simulating a cell using model-blending approach.

matrix. We have an equivalent input–output relationship between $\mathbf{u}_k$ and $\mathbf{y}_k$ if we write

$$\mathbf{x}_{k+1}^{(1)} = \underbrace{\left(\mathbf{T}^{(1)}\right)^{-1}\mathbf{A}^{(0)}\mathbf{T}^{(1)}}_{\mathbf{A}^{(1)}}\mathbf{x}_k^{(1)} + \underbrace{\left(\mathbf{T}^{(1)}\right)^{-1}\mathbf{B}^{(0)}}_{\mathbf{B}^{(1)}}\mathbf{u}_k$$

$$\mathbf{y}_k = \underbrace{\mathbf{C}^{(0)}\mathbf{T}^{(1)}}_{\mathbf{C}^{(1)}}\mathbf{x}_k^{(1)} + \mathbf{D}\mathbf{u}_k.$$

We have great freedom in choosing the transformation matrix $\mathbf{T}^{(1)}$, so long as it is invertible.

Consider first choosing $\mathbf{T}^{(1)} = \mathbf{V}$, where $\mathbf{V}$ is a matrix whose columns are the eigenvectors of $\mathbf{A}^{(0)}$. The resulting $\mathbf{A}^{(1)}$ matrix will be diagonal.[3] The diagonal elements of $\mathbf{A}^{(1)}$ are called the *poles* of the system, and represent the dynamic time constants. Already, this first transformation has uncovered some physical meaning in the model. Also, the storage requirements of the transformed model are reduced, as the $n \times n$ $\mathbf{A}^{(1)}$ matrix will contain only n non-zero values (on its diagonal), and computation requirements will be reduced, since only the diagonal elements of the $\mathbf{A}$ matrices need to be blended and fewer multiplies are needed to implement $\mathbf{A}^{(1)}\mathbf{x}_k^{(1)}$ than to implement $\mathbf{A}^{(0)}\mathbf{x}_k^{(0)}$.

Since eigenvectors are unique only up to a scaling factor, we can utilize this remaining degree of freedom to simplify our matrices further. Here, we elect to normalize the $\mathbf{B}$ matrix to have units value elements. This, of course, presupposes that there are no zero elements in $\mathbf{B}$, which is guaranteed if the system is completely controllable. The Ho–Kalman algorithm used in the DRA always produces a minimal state-space description, so we have this guarantee.

Thus, we then apply a second transformation, choosing $\mathbf{T}^{(2)} = \text{diag}(\mathbf{B}^{(1)})$. In the resulting transformed model, $\mathbf{B}^{(2)}$ contains only ones, and $\mathbf{A}^{(2)}$ will be unchanged from $\mathbf{A}^{(1)}$. This transformation has resulted in $\mathbf{B}^{(2)}$ and $\mathbf{C}^{(2)}$ matrices that are all scaled in a consistent way. It also reduces storage requirements, as $\mathbf{B}^{(2)}$ is known to always contain only ones, which do not need to be stored. Computation has also been reduced, as the $\mathbf{B}$ matrices do not need to be blended (they are all the same), and because the multiplication $\mathbf{B}_k\mathbf{u}_k$ is simply a repetition of the elements of $\mathbf{u}_k$, without multiplication.

Finally, we choose a third transformation matrix $\mathbf{T}^{(3)}$ to permute the elements of $\mathbf{A}^{(2)}$ such that $\mathbf{A}^{(3)}$ remains diagonal, but its elements appear in order of ascending magnitude. The $\mathbf{B}^{(3)}$ matrix remains all ones. For any particular temperature and SOC setpoint, we define the final scaled and sorted precomputed model as having $\mathbf{A} = \mathbf{A}^{(3)}$, $\mathbf{B} = \mathbf{B}^{(3)} = 1_{n\times 1}$, and $\mathbf{C} = \mathbf{C}^{(3)}$. The model $\mathbf{D}$ matrix is unchanged from the one produced by the DRA.

4.3. Stability of the blended model

Another concern that must be addressed is whether the model-blending method guarantees a stable time-varying model. It is intuitively appealing that linear time-varying combinations of stable non-time-varying systems should be stable, but this is not always the case, depending on how the systems are connected.

With the method we propose, the model blending will always result in a stable system. To see this, consider the following. The blended model is computed using a weighted sum of the four nearest DRA matrices using bilinear interpolation. The state-space model can be written

$$\mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k$$

$$\mathbf{y}_k = \mathbf{C}_k\mathbf{x}_k + \mathbf{D}_k\mathbf{u}_k$$

where $\mathbf{A}_k$ is diagonal with diagonal entries $0 \le a_{ii} < 1$ except for the integrator pole where $a_{ii} = 1$. The $\mathbf{B}_k$ matrix of each ROM is $[1 \quad 1 \quad \ldots \quad 1]^T$. The state vector, $\mathbf{x}_k$, can be found as

$$\mathbf{x}_k = \left(\prod_{j=0}^{k-1}\mathbf{A}_{k-1-j}\right)\mathbf{x}_0 + \sum_{i=0}^{k-1}\left(\prod_{j=0}^{k-i-2}\mathbf{A}_{k-1-j}\right)\mathbf{B}_i\mathbf{u}_i$$

where $\mathbf{A_k} \equiv \mathbf{I}$ for $k < 0$. Using the triangle inequality, the infinity norm of the state vector is

$$\|\mathbf{x}_k\|_\infty \le \left\|\left(\prod_{j=0}^{k-1}\mathbf{A}_{k-1-j}\right)\mathbf{x}_0\right\|_\infty + \left\|\sum_{i=0}^{k-1}\left(\prod_{j=0}^{k-i-2}\mathbf{A}_{k-1-j}\right)\mathbf{B}_i\mathbf{u}_i\right\|_\infty . \tag{7}$$

In order to demonstrate bounded-input bounded-output (BIBO) stability, we require that the input be bounded: $\|\mathbf{u}_k\|_\infty < \gamma < \infty$, where γ is some arbitrary finite bounding value.

Note that by the operations performed in Section 4.2, the diagonalization of the $\mathbf{A}_k$ matrix has caused the individual states to become decoupled. In particular, every model will have the exact same integrator pole value where $a_{ii} = 1$, which does not then require blending between models. To ensure that the time-varying model is stable, we need look only at the non-integrator dynamics, where $0 \le a_{ii} < 1$.

To show that a bounded input always results in a bounded output, we look at the terms on the right-hand-side of Equation (7). Taking advantage of the fact that we have diagonalized all $\mathbf{A}_k$ matrices, the first term is bounded by

[3] For this to be possible, the $\mathbf{A}^{(0)}$ matrix must be diagonalizable, requiring that the eigenvectors in $\mathbf{V}$ be linearly independent. In our experience, the output of the DRA has always resulted in a diagonalizable $\mathbf{A}^{(0)}$ matrix; however, we know of no guarantee of this. In cases where the eigenvectors are linearly dependent, it is always possible to choose a transformation to put the $\mathbf{A}^{(1)}$ matrix into a Jordan form, which is what should be done instead [19].

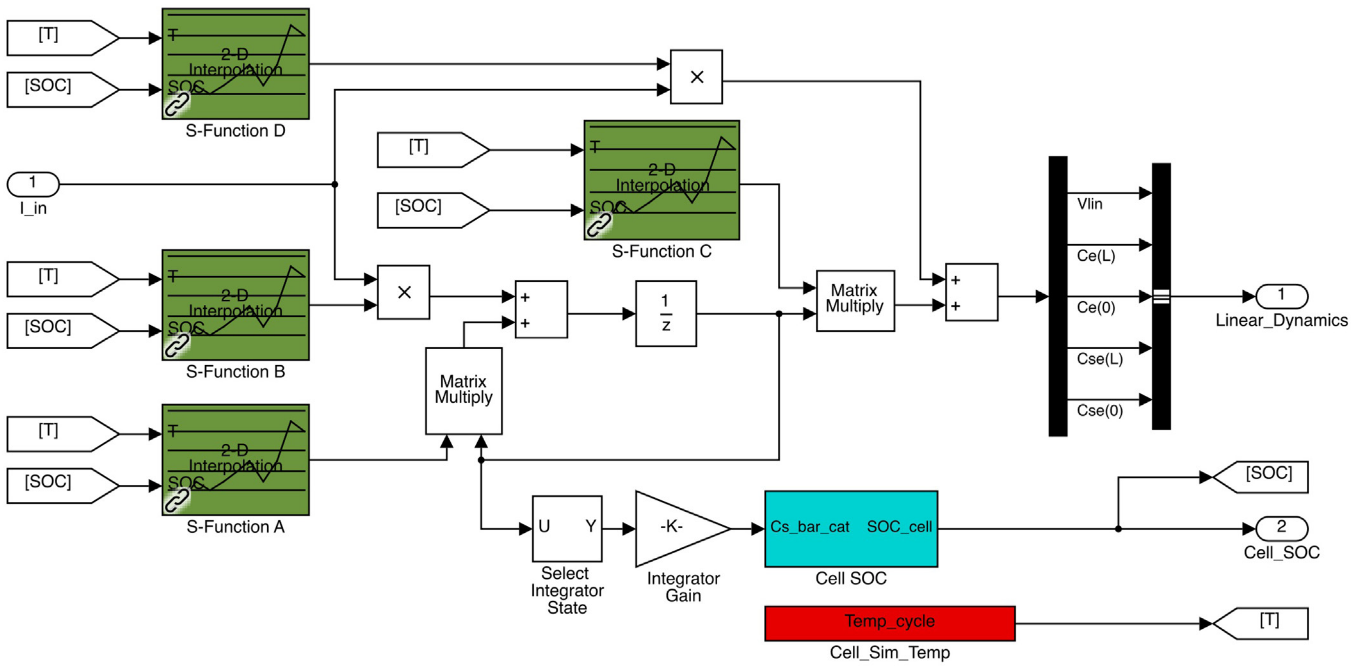


Fig. 5. An implementation of the model-blending approach using Simulink.

$$\left\|\left(\prod_{j=0}^{k-1}\mathbf{A}_{k-1-j}\right)\mathbf{x}_0\right\|_\infty \le (\max_a)^k\|\mathbf{x}_0\|_\infty,$$

where $\max_a = \max_{i,k}(a_{k,ii}) < 1$. Thus, this term is always finite if $\|\mathbf{x}_0\|_\infty < \infty$. Turning to the second term on the right-hand-side of Equation (7) we have,

$$\left\|\sum_{i=0}^{k-1}\left(\prod_{j=0}^{k-i-2}\mathbf{A}_{k-1-j}\right)\mathbf{B}_i\mathbf{u}_i\right\|_\infty \le \sum_{i=0}^{k-1}\left\|\left(\prod_{j=0}^{k-i-2}\mathbf{A}_{k-1-j}\right)\right\|_\infty \|\mathbf{B}_i\mathbf{u}_i\|_\infty. \tag{8}$$

Because all of the values in the **A** matrix are less than 1, we can write

$$\left\|\left(\prod_{j=0}^{k-i-2}\mathbf{A}_{k-1-j}\right)\right\|_\infty \le (\max_a)^{k-i-1}.$$

Equation (8) is simplified to

$$\left\|\sum_{i=0}^{k-1}\left(\prod_{j=0}^{k-i-2}\mathbf{A}_{k-1-j}\right)\mathbf{B}_i\mathbf{u}_i\right\|_\infty \le \gamma(\max_a)^{k-1}\sum_{i=0}^{k-1}\left(\frac{1}{\max_a}\right)^i.$$

The formula for the geometric series is used to give

Table 2
Cell parameters for simulation.

Symbol	Units	Negative electrode	Separator	Positive electrode
L	μm	128	76	190
R_s	μm	12.5	–	8.5
A	m^2	1	1	1
σ	$S\ m^{-1}$	100	–	3.8
ε_s	$m^3\ m^{-3}$	0.471	–	0.297
ε_e	$m^3\ m^{-3}$	0.357	0.724	0.444
brug	–	1.5	–	1.5
$c_{s,\max}$	mol m^{-3}	26,390	–	22,860
$c_{e,0}$	mol m^{-3}	2000	2000	2000
$\theta_{i,\min}$	–	0.05	–	0.78
$\theta_{i,\max}$	–	0.53	–	0.17
D_s	$m^2\ s^{-1}$	3.9×10^{-14}	–	1.0×10^{-13}
D_e	$m^2\ s^{-1}$	7.5×10^{-11}	7.5×10^{-11}	7.5×10^{-11}
t_+^0	–	0.363	0.363	0.363
k	$mol^{-1/2}\ m^{5/2}\ s^{-1}$	1.94×10^{-11}	–	2.16×10^{-11}
α	–	0.5	–	0.5
R_{film}	$\Omega\ m^2$	0.0	–	–

We compute $\sigma^{\text{eff}} = \sigma\varepsilon_s$, $\kappa^{\text{eff}} = \kappa\varepsilon_e^{\text{brug}}$, $D_e^{\text{eff}} = D_e\varepsilon_e^{\text{brug}}$.
In the electrolyte, conductivity is a function of concentration:
$\kappa(c_e) = 4.1253 \times 10^{-2} + 5.007 \times 10^{-4}c_e - 4.7212 \times 10^{-7}c_e^2 + 1.5094 \times 10^{-10}c_e^3 - 1.6018 \times 10^{-14}c_e^4$.
For the negative electrode, the open-circuit potential function is:
$U_{\text{ocp}}(\theta) = -0.16 + 1.32\exp(-3.0\theta) + 10.0\exp(-2000.0\theta)$.
For the positive electrode, the open-circuit potential function is:
$U_{\text{ocp}}(\theta) = 4.19829 + 0.0565661\tanh(-14.5546\theta + 8.60942) - 0.0275479\left[\frac{1}{(0.998432-\theta)^{0.4924656}} - 1.90111\right] - 0.157123\exp(-0.04738\theta^6) + 0.810239\exp[-40(\theta - 0.133875)]$.

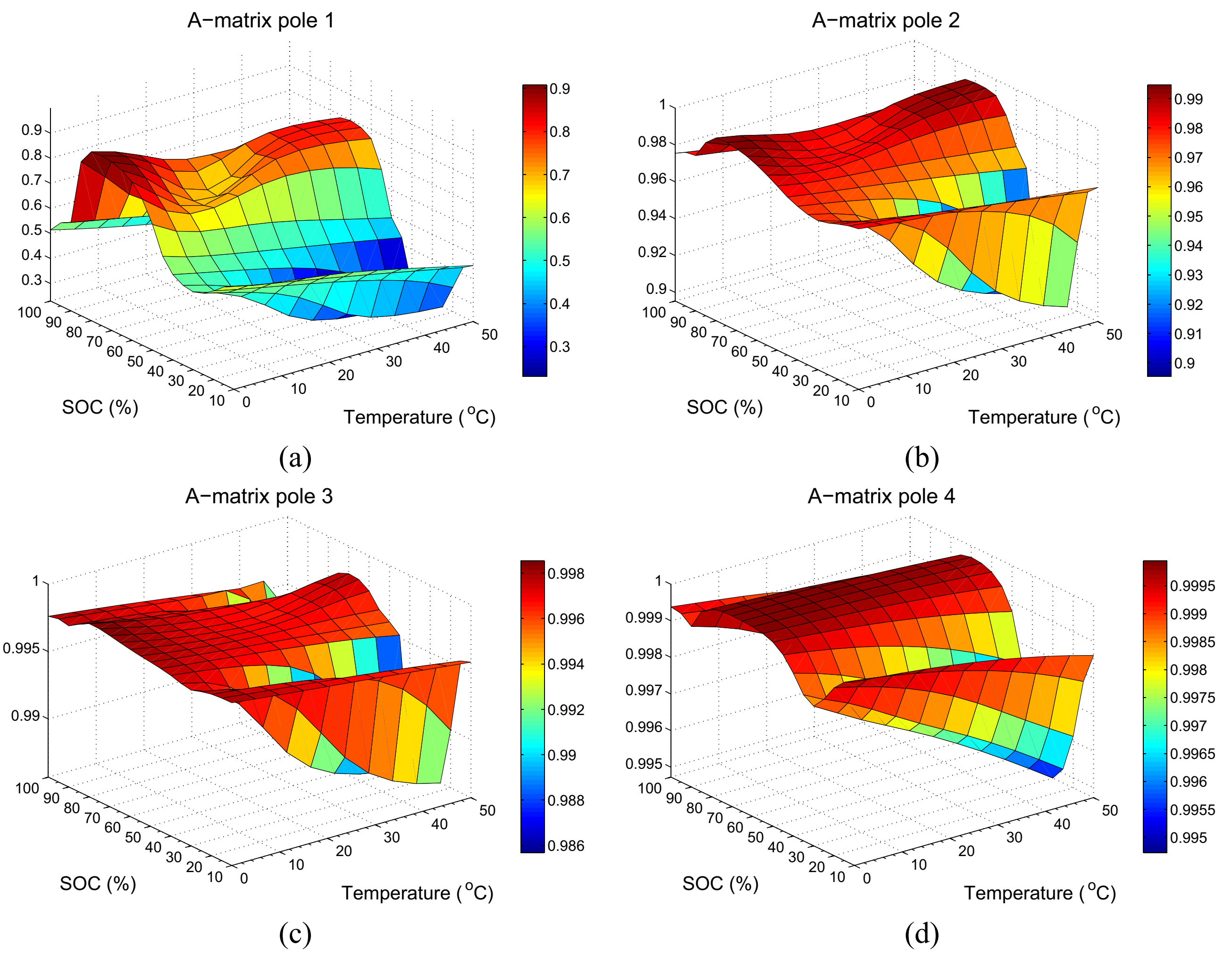


Fig. 6. Discrete time system poles.

$$\left\|\sum_{i=0}^{k-1}\left(\prod_{j=0}^{k-i-2}\mathbf{A}_{k-1-j}\right)\mathbf{B}_i\mathbf{u}_i\right\|_\infty \le \gamma(\max_a)^{k-1}\left[\frac{1-\left(\frac{1}{\max_a}\right)^k}{1-\left(\frac{1}{\max_a}\right)}\right]$$
$$\le \gamma\left[\frac{(\max_a)^{k-1}-\left(\frac{1}{\max_a}\right)}{1-\frac{1}{\max_a}}\right] < \infty.$$

Both terms on the right hand side of Equation (7) are finite. Therefore, $\|\mathbf{x}_k\|_\infty < \infty$.

The infinity norm of the output equation is defined as

$$\|\mathbf{y}_k\|_\infty = \|\mathbf{C}_k\mathbf{x}_k + \mathbf{D}_k\mathbf{u}_k\|_\infty.$$

In order to show BIBO stability, this norm must be finite. Using the triangle inequality gives,

$$\|\mathbf{y}_k\|_\infty \le \|\mathbf{C}_k\mathbf{x}_k\|_\infty + \|\mathbf{D}_k\mathbf{u}_k\|_\infty \le \|\mathbf{C}_k\|_\infty\|\mathbf{x}_k\|_\infty + \|\mathbf{D}_k\|_\infty\|\mathbf{u}_k\|_\infty.$$

Since $\|\mathbf{x}_k\|_\infty < \infty$ and the $\mathbf{C}_k$ and $\mathbf{D}_k$ matrices are finite, $\|\mathbf{y}_k\|_\infty < \infty$ which shows BIBO stability.

4.4. Implementation

Various different implementations of the model-blending approach are possible. We have implemented the model in MATLAB, and also in Simulink [20,21]. The latter is particularly useful, because the Simulink Coder can automatically produce embedded C and C^{++} code, which can be implemented in an embedded battery-management system, without the user needing to write any low-level code (and debug same) [22].

Fig. 5 shows a Simulink block-diagram subsystem corresponding to the "Make Blended Model" and "Simulate Blended Model" blocks of Fig. 4. The temperature profile for the entire simulation is input from the MATLAB workspace via the "Temp_cycle" vector. The temperature value at every point in time is distributed throughout the subsystem via the "GoTo" tag "[T]". SOC is dynamically computed from the model state-vector integrator state, and is provided to the rest of the model via the "GoTo" tag "[SOC]". Cell applied current is an input to this subsystem via the "I_in" port, and outputs from the subsystem comprise the linear output vector $\mathbf{y}_k$ via the "Linear_Dynamics" output port, and SOC via the "Cell_SOC" output port.

For speed and for a seamless conversion to embedded C code via the Simulink Coder, the bilinear interpolation was written as a compiled C-language Simulink "S Function". These are shown in the

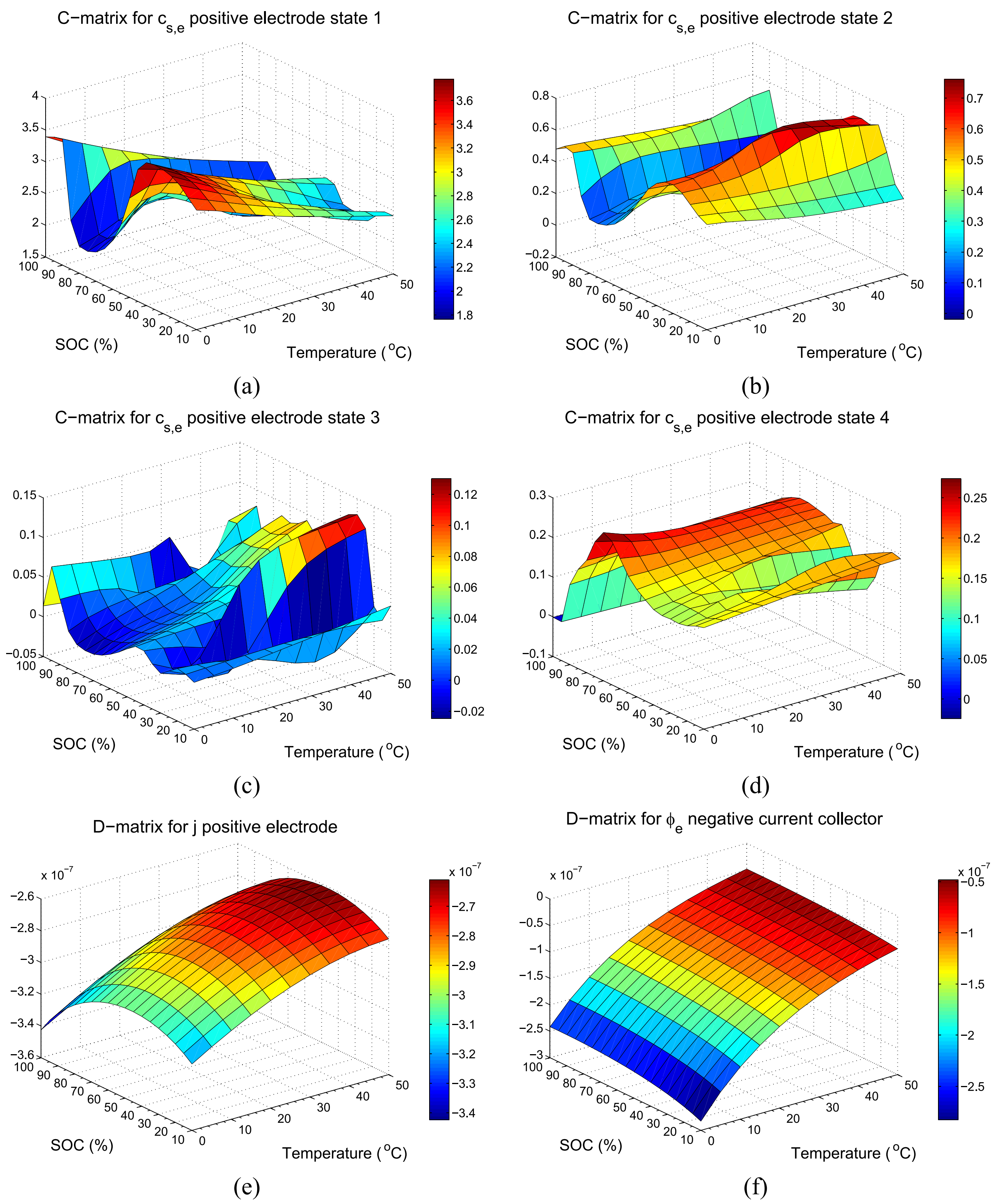


Fig. 7. Example of typical **C**-matrix and **D**-matrix values.

block diagram as the blocks “S-Function A,” “S-Function B,” “S-Function C,” and “S-Function D.” (Note that the interpolation function is not needed for the $\mathbf{B}_k$ matrix, but this diagram includes it for generality.) These interpolation blocks compute the time-varying $\mathbf{A}_k$, $\mathbf{B}_k$, $\mathbf{C}_k$, and $\mathbf{D}_k$, as previously described, from the input values of temperature and SOC.

The remainder of the block diagram implements the model Equations (5) And (6). Note that the block labeled $1/z$ is a unit-delay operator—the signal at the input to this delay is $\mathbf{x}_{k+1}$, and the signal at its output is $\mathbf{x}_k$. The bus generation blocks at the top-right of the diagram give names and a consistent ordering to signals, so that the output of the subsystem may be used by other blocks in a Simulink implementation.

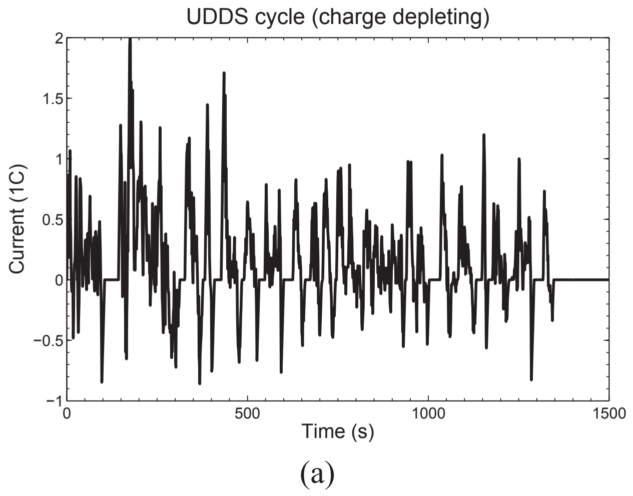


Fig. 8. Input current profiles.

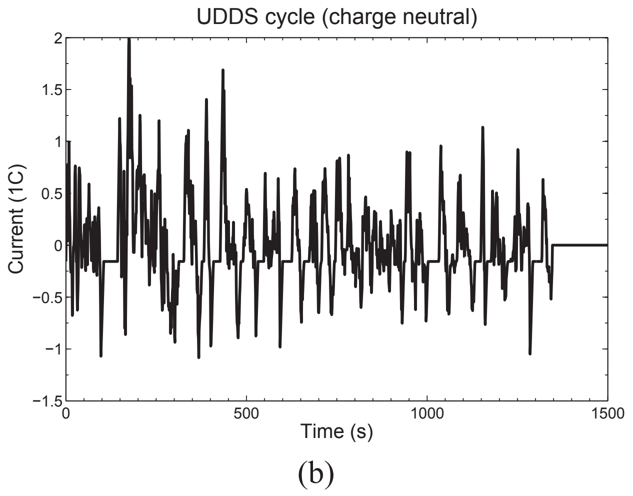


Table 3
RMS error across SOC and temperature.

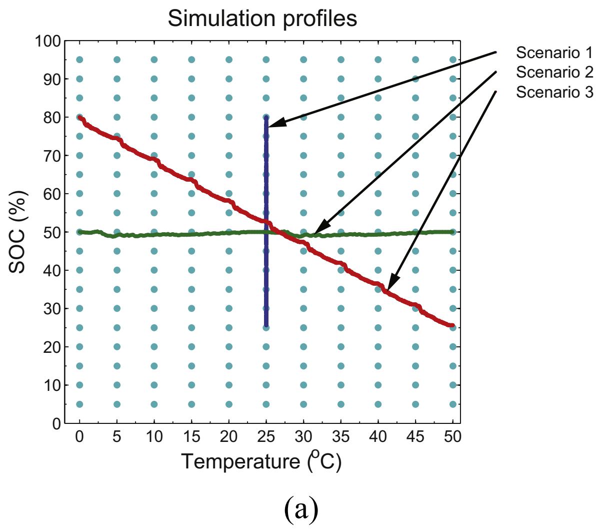
	0 °C	10 °C	20 °C	30 °C	40 °C	50 °C
5%	5.44	4.20	3.55	3.03	2.59	2.19
15%	9.27	4.32	2.01	1.13	0.91	0.87
25%	8.92	4.07	1.80	0.82	0.58	0.57
35%	8.63	3.92	1.69	0.78	0.59	0.58
45%	8.88	4.05	1.82	0.95	0.71	0.66
55%	8.93	4.13	1.80	0.91	0.71	0.67
65%	8.97	4.24	1.88	0.90	0.82	0.80
75%	9.20	4.28	1.93	0.92	0.74	0.69
85%	9.72	4.63	2.10	0.97	0.59	0.50
95%	9.76	4.60	2.16	1.09	0.71	0.49

While this paper concerns itself with modeling a single cell, many applications require large battery packs having many cells. Often, there will be significant temperature and SOC differences between cells in a large battery pack. In such a situation, the most accurate implementation approach would be to propagate separate models for every cell, where individual time-varying $\mathbf{A}_k$, $\mathbf{B}_k$, $\mathbf{C}_k$, and $\mathbf{D}_k$ matrices are computed based on each cell's SOC and temperature, and where individual state vectors $\mathbf{x}_k$ are also propagated for each cell. In that case, the subsystem in Fig. 5 would be replicated for every cell.

Additionally, this paper considers model blending as a function of SOC and temperature only. In principle, it is possible to blend models based on other factors as well (e.g., some metric of cell age). We have not pursued this idea thoroughly at this point, and do mention that increasing the dimensionality of the setpoint dramatically increases the storage requirements for the set of individual setpoint **A**, **B**, **C**, and **D** matrices. There may well be better approaches to capturing aging effects, as they happen over a much slower timescale than changes in SOC and temperature.

5. Results

We now present some results to demonstrate the effectiveness of model blending. In all cases, results presented in this section



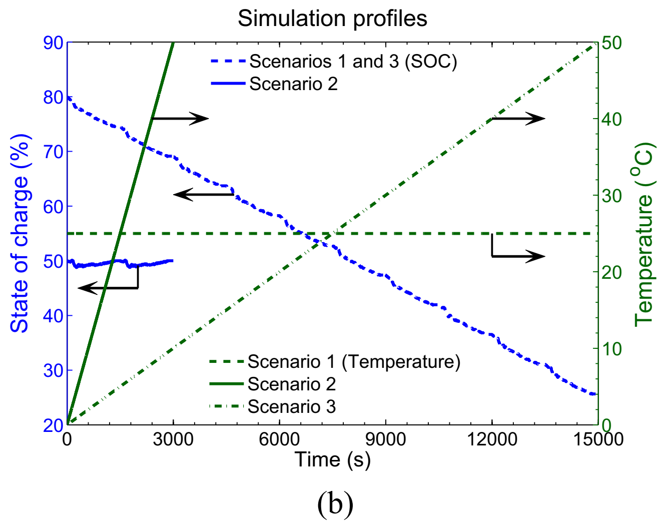


Fig. 9. SOC and temperature profiles during the simulations used in the results: (a) parametric plot showing paths through grid of ROMs; (b) standard plot of SOC and temperature versus time.

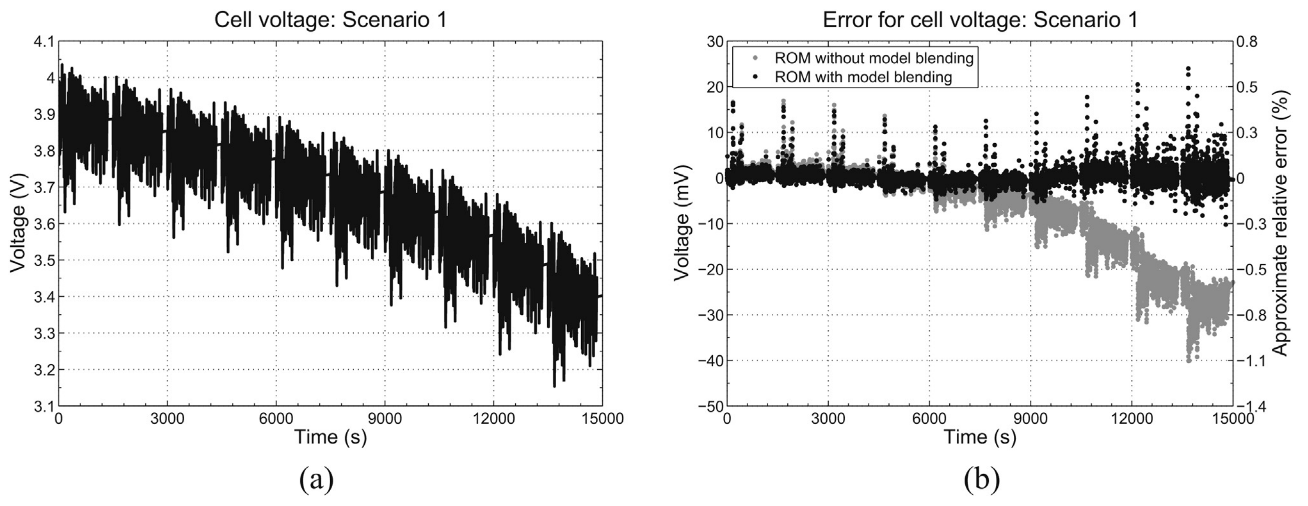


Fig. 10. Cell voltage results for the wide SOC simulation.

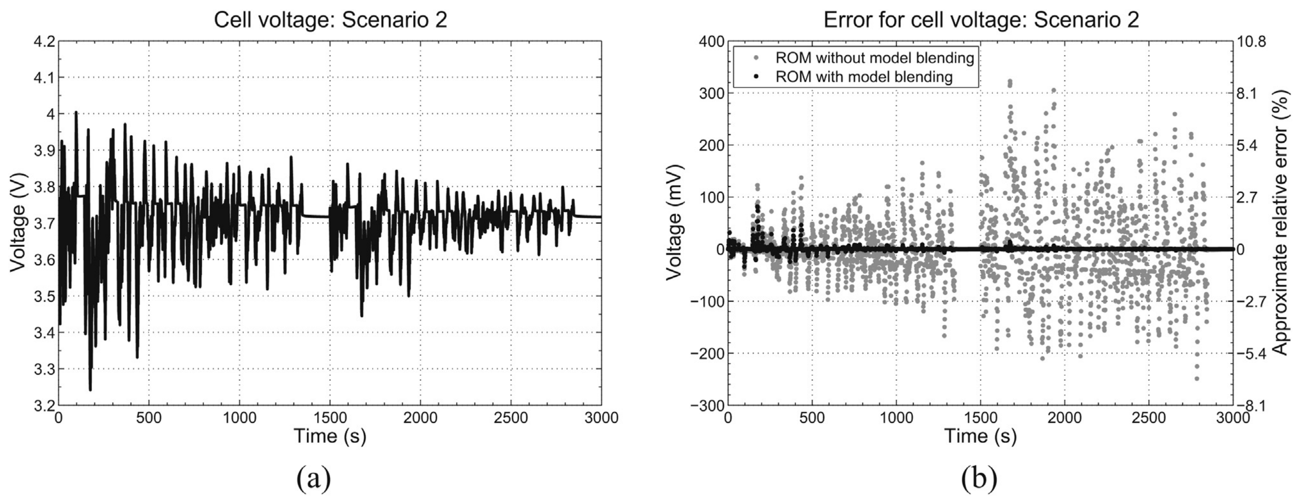


Fig. 11. Simulation of temperature change.

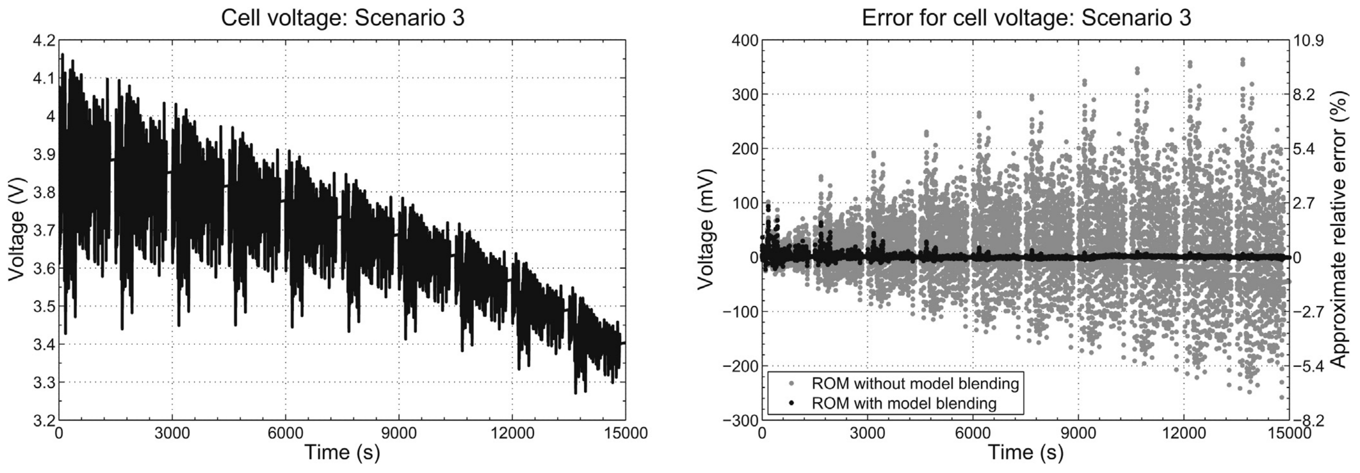


Fig. 12. Cell voltage results for the wide SOC and temperature simulation.

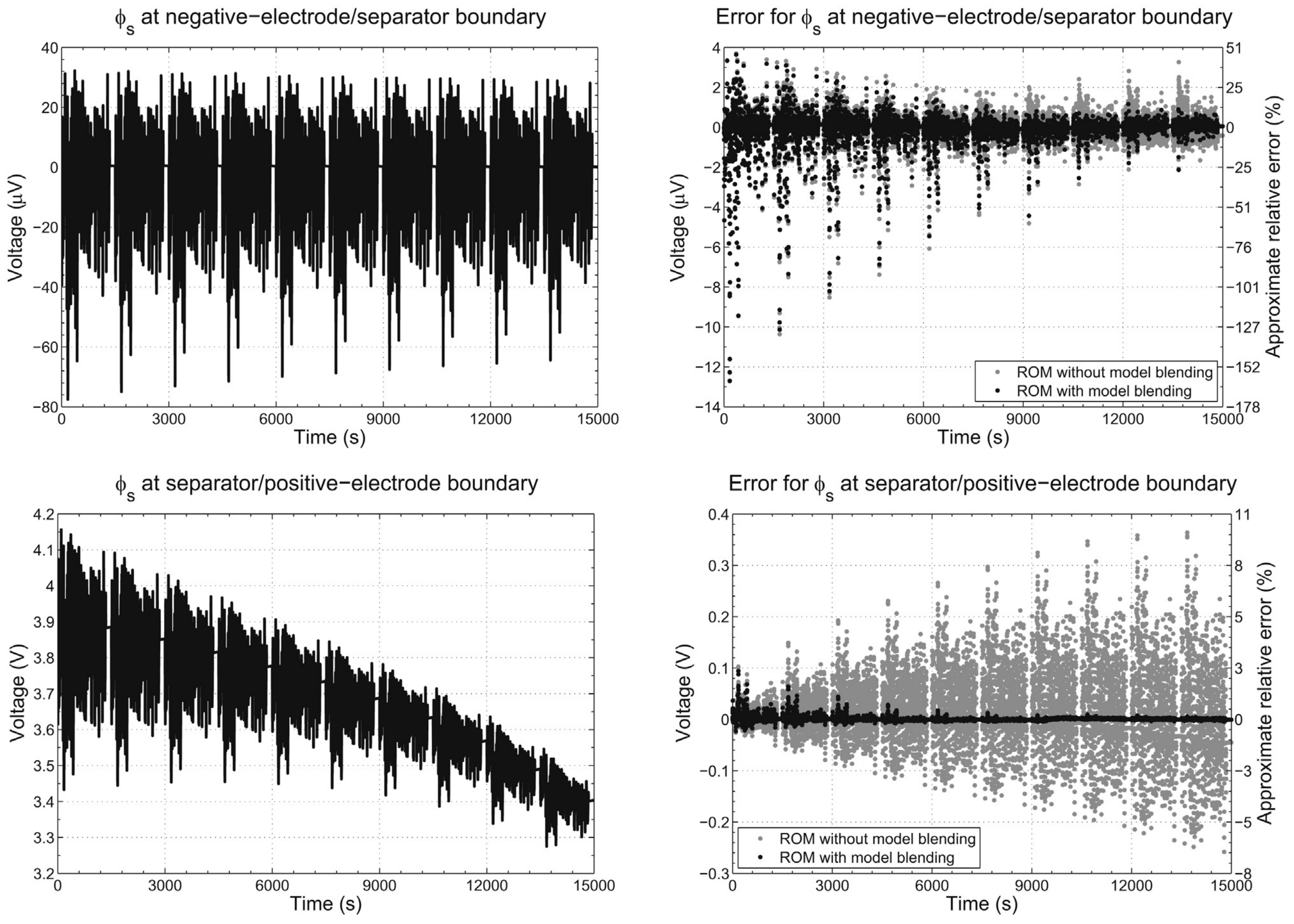


Fig. 13. Φ_s results for the wide SOC and temperature simulation.

were generated using the cell parameters in Refs. [23,24], which are shown in Table 2. The DRA was used to generate a fifth-order ROM for every SOC between 5% and 100% in 5% increments and for every temperature between 0 °C and 50 °C in 5 C° increments. As discussed in Ref. [12], this process is entirely automated and hands free. The setpoints are illustrated in Fig. 9 as a matrix of cyan dots (in the web version). For each ROM, the six temperature-dependent parameters, $D_{s,\text{neg}}$, $D_{s,\text{pos}}$, D_e, κ, k_{neg}, and k_{pos}, are modified according to Equation (3) using the activation energies given in Table 1.

As described in Ref. [12], the DRA requires four tuning parameters to produce each ROM. These are the high-rate sampling frequency, the size of the Hankel matrix, the number of states in the reduced-order system, and the sampling period of the final discrete-time reduced order model. The high-rate sampling frequency was chosen to be 128 Hz, the Hankel matrix block size was 8000, the order of the system was five, and the sampling period of the final discrete-time system was 1 s. (These are identical to the values used in Ref. [12] and a detailed explanation behind these parameter settings is given in the same paper.) As a point of reference, each set of **A**, **B**, **C** and **D** matrices comprising an individual ROM is computed off-line in about 6 min 45 s on a 64 GB Intel Xeon X5650 running at 2.67 GHz under Windows 7 (64 bit). These matrices are computed one time only via the process depicted in Fig. 1, then used on-line in the process depicted in Fig. 4, where the computations are easily performed more quickly than real time, even on an inexpensive microcontroller.

5.1. Smoothness of the model matrices versus SOC and temperature

For the model-blending approach to work, the state-space matrices must have smoothly changing parameters over the expected SOC and temperature range. Then, blending model values via interpolation between setpoints for any specific operational SOC and temperature will compute a numeric result that is close to what would have been computed if a dedicated ROM had been precomputed at that SOC and temperature via the DRA. In effect, the smoothness (or lack thereof) of the model parameters dictates the number of precomputed setpoint ROMs needed to be able to accurately capture the changing dynamics of the model when model blending.

We now show the setpoint model matrices to vary in a sufficiently smooth manner. For each setpoint ROM, the diagonal **A** matrix has one integrator term (with “pole” equal to 1.0 in every case); the value of the remaining four “poles” are plotted in Fig. 6(a)–(d) as functions of SOC and temperature. Since the **A** matrix has its poles sorted in ascending order, the first pole represents the fastest dynamics, and the fourth pole represents the slowest non-integrator dynamics. We see that all four poles demonstrate smooth variation, such that we would expect model blending to yield good intermediate models in-between stored precomputed models.

Fig. 7(a)–(d) shows similar representative samples from the **C** matrix (corresponding to $c_{s,e}$ in the positive electrode) and Fig. 7(e) and (f) shows **D** matrix parameters for j and ϕ_e. For the most part, these variations are smooth as well. The least smooth result is

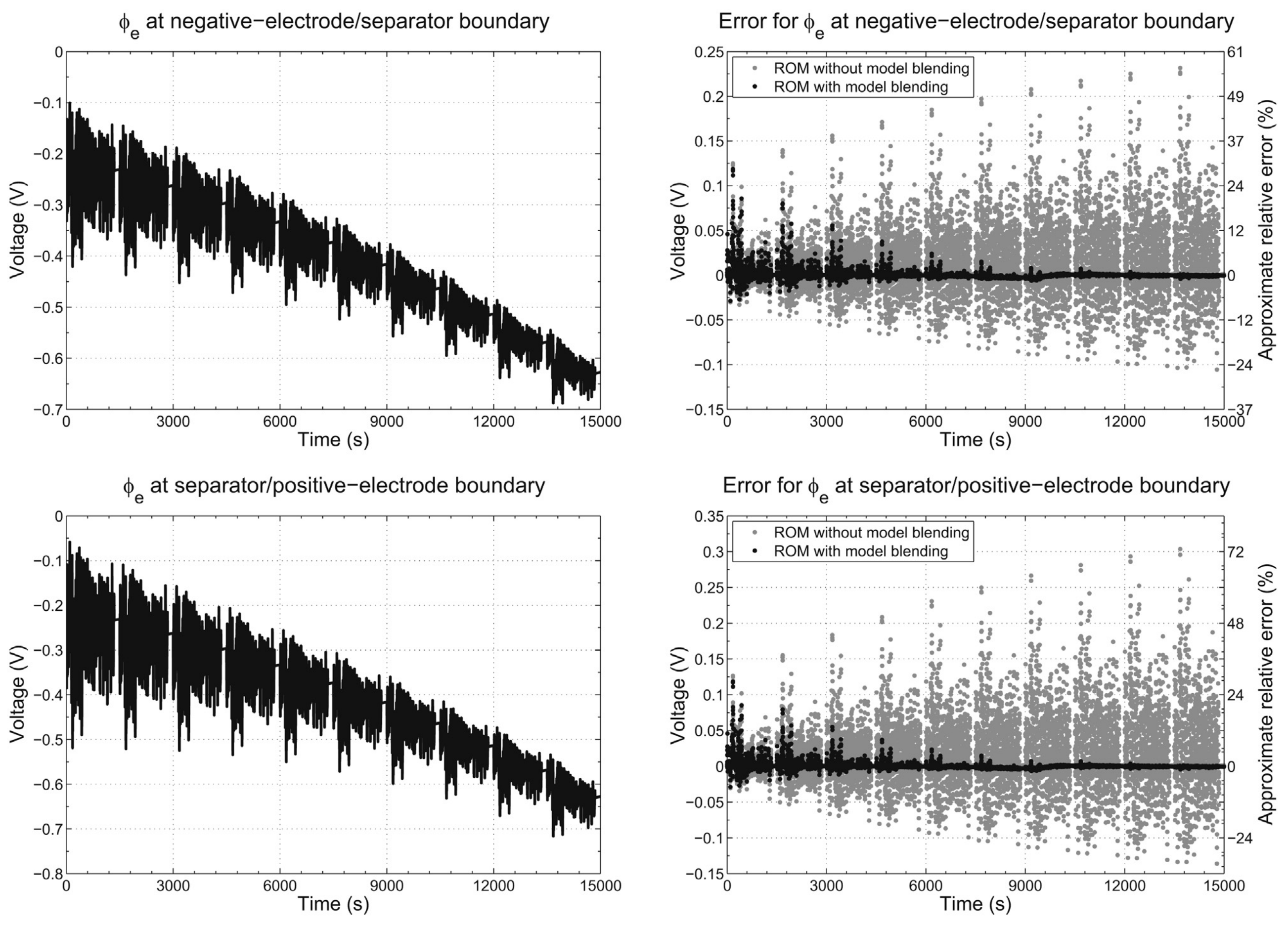


Fig. 14. Φ_e results for the wide SOC and temperature simulation.

Fig. 7(c), however we will later see in Fig. 15 that predictions using this model variable are quite good in its non-smooth region, so we are satisfied with it. Had results not turned out so well, we could have added more model setpoints in the region of least smoothness, to be able to better capture the high-frequency variation.[4]

5.2. Drive-cycle simulations

To demonstrate the effectiveness of model blending over wide ranges of temperature and SOC, we simulate the above-mentioned cell using a finite-element PDE solver and using the model-blended ROM. The cell input current for the simulations is based on the Environmental Protection Agency's (EPA) Urban Dynamometer Driving Schedule (UDDS), which was developed to represent city driving conditions for light duty vehicles. Fig. 8(a) shows the normalized UDDS profile of current versus time, as computed for a medium-size electric vehicle. The maximum absolute rate is 2C, which for this cell corresponds to 41 A.

The UDDS cycle in Fig. 8(a) is charge depleting, reducing cell SOC by about 5.5%. For cases where we wanted to investigate the effect of varying temperature without large changes in SOC, we developed a modified charge-neutral UDDS profile that retains the same peak current range but which has an equal amount of charge and discharge. The modified UDDS cycle was produced by calculating the total discharge from the UDDS cycle and subtracting the appropriate equal amount from each time step to produce a charge-neutral version. A multiplier was then applied to leave the maximum current for the profile unchanged. The charge-neutral UDDS cycle is shown in Fig. 8(b).

For reference purposes, we conducted non-model-blended simulations using the charge-neutral UDDS profile and constant temperature at SOC and temperature values corresponding to every model setpoint. The "truth" result is generated using PDE simulations of the pseudo-2D porous-electrode model, which we implemented in COMSOL Multiphysics version 4.3 [25]. Table 3 lists the root-mean-squared (RMS) cell-voltage prediction error between the PDE result and the non-model-blended ROM result for all cases. Since the individual DRA ROMs are optimized for charge-neutral constant-temperature conditions, these simulation results give a best-case lower bound on expected model-blended ROM RMS error when moving between and among the setpoints.

The following three subsections present results for three different simulation scenarios to provide insight into the impact of changing SOC only, changing temperature only, and changing both SOC and temperature together. In the first case, the cell is run through ten consecutive charge-depleting UDDS cycles and the temperature is held constant at 25 °C. For the second case the input is two cycles of the charge-neutral UDDS cycle while the temperature changes linearly from 0 °C to 50 °C during the

[4] This brings up the interesting question of "what is the optimal minimal set of precomputed setpoints?" The answer to this question is presently open, and is beyond the scope of this work.

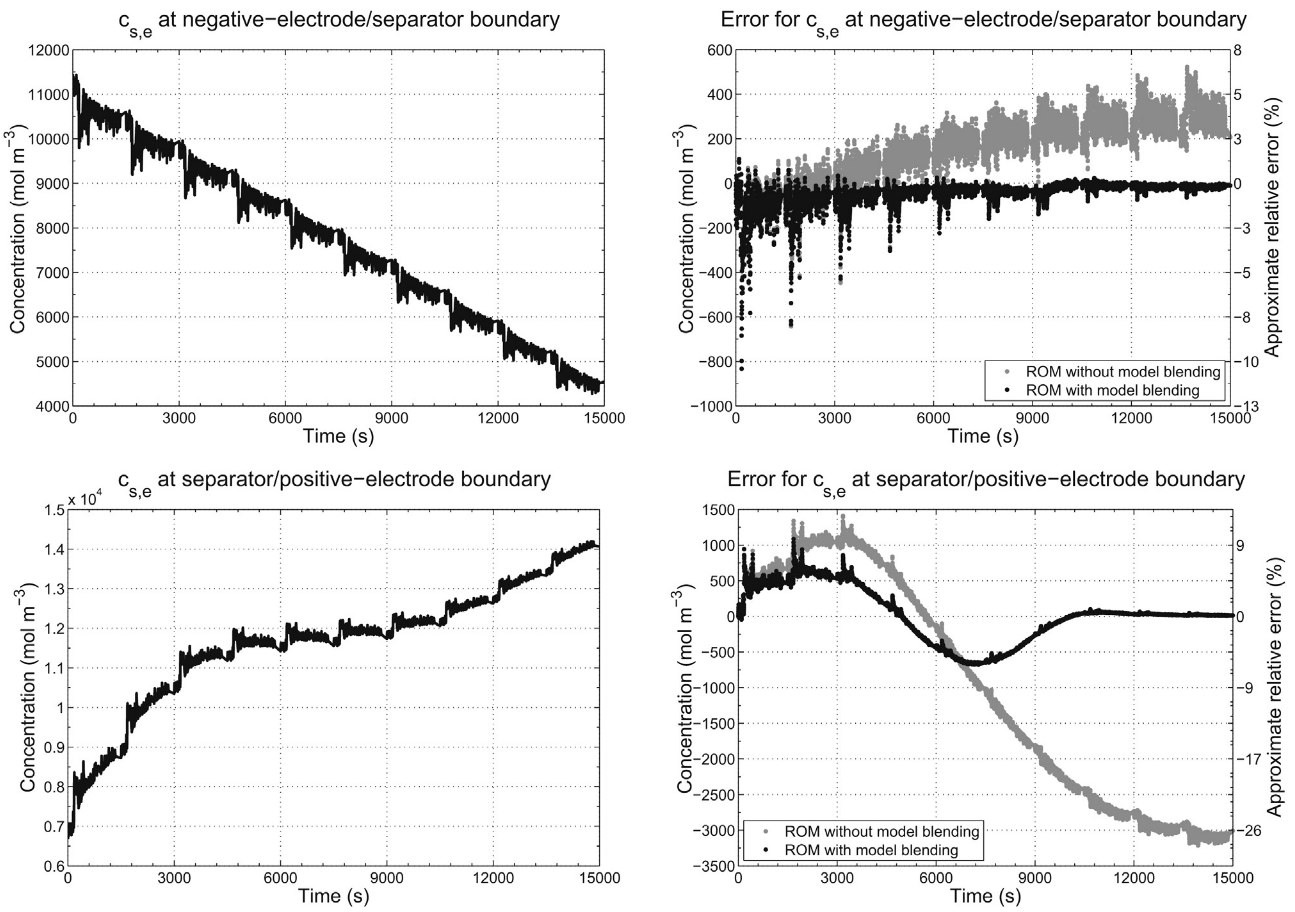


Fig. 15. $c_{s,e}$ results for the wide SOC and temperature simulation.

3000 s simulation. In the final case, both the SOC and the temperature vary throughout the simulation. SOC and temperature versus time for these three different scenarios are shown as parametric plots in Fig. 9(a), where the cyan dots represent the location of each setpoint ROM and the lines indicate the profile of the SOC and temperature for each of the three simulations, and as standard time plots in Fig. 9(b).

5.3. Scenario 1: wide SOC range, constant temperature

We first illustrate the performance of the blended model by running ten consecutive UDDS cycles at a constant temperature of 25 °C. During the simulation, the cell discharges from 80% to about 25% SOC. This is approximately consistent with the range used by the battery pack in the Chevy Volt [26]. Fig. 10(a) depicts the cell voltage calculated using the full-order PDE model implemented in COMSOL. Fig. 10(b) shows (on the left axis) the error in the cell voltage calculation both for the blending and non-blending case. The results for the ROM without model blending are generated using the singe ROM linearized around 80% SOC. The cell voltage for the ROM becomes less accurate as the simulation progresses and the cell SOC changes significantly from the linearization setpoint. The simulation using the model-blended ROM has a cell voltage RMS error of 1.60 mV while the non-blended model has a 12.1 mV error. Referring back to Table 3, the blended RMS error of 1.60 mV is a very reasonable value to expect when traversing the table from 80% to 25% SOC at 25 °C.

Fig. 10(b) and many other figures to follow also show (on the right axis) an approximate relative error. This error is calculated as

$$\text{approximate relative error} = \frac{\text{truth} - \text{estimate}}{\text{average}(|\text{truth}|)} \times 100(\%).$$

The denominator of this fraction is modified from a more standard definition where

$$\text{relative error} = \frac{\text{truth} - \text{estimate}}{\text{truth}} \times 100(\%)$$

to avoid division by zero for those quantities that could be zero at some points in time.

5.4. Scenario 2: near-constant SOC, wide temperature range

In this section we examine the impact of temperature on the cell performance, independent of the cell SOC. Two charge-neutral UDDS cycles are used as input and the temperature increases linearly from 0 °C to 50 °C over the 3000 s simulation. The initial cell SOC is 50% and throughout the simulation the SOC deviates by a maximum of 1% from the starting value before returning to 50% by the end of the simulation. The ROM results without model blending use the single setpoint ROM generated at 50% SOC and 0 °C throughout the simulation. Fig. 11(a) shows the COMSOL results of the full-order PDE model and Fig. 11(b) shows the error for both the blended and non-blended approach. As expected, both approaches produce similar output during the early part of the simulation. However, as the temperature changes, the model-blended ROM accurately tracks the rigorous PDE while the non-blended single ROM model does not. The RMS error for the single ROM simulation is 64.4 mV, while the RMS error using the model-blending approach is 4.6 mV. Referring back to

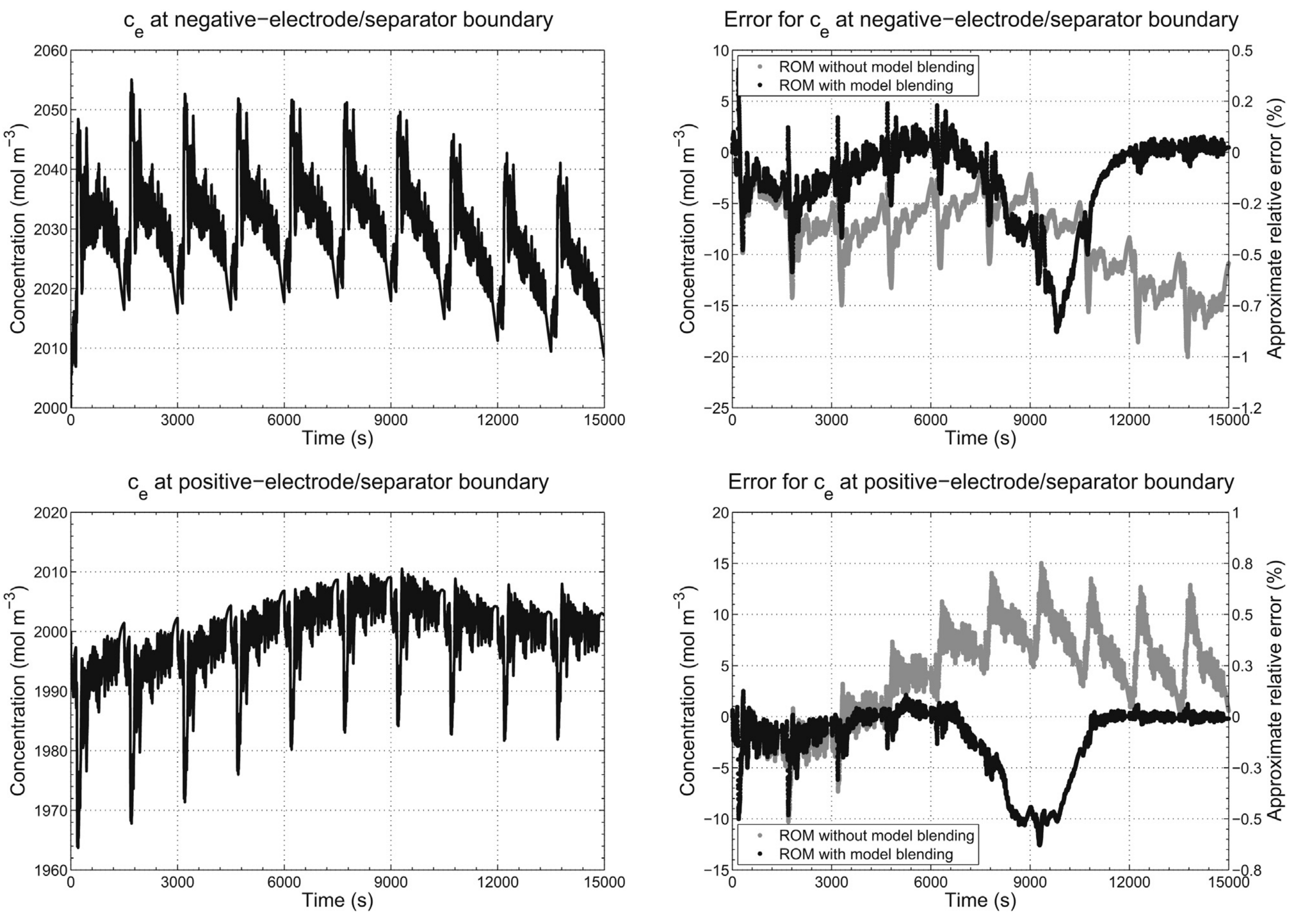


Fig. 16. c_e results for the wide SOC and temperature simulation.

Table 3, the blended RMS error of 4.6 mV is a very reasonable value to expect when traversing the table from 0 °C to 50 °C at 50% SOC: it is better than if the temperature remained at 0 °C, but not as good as if the temperature were warmer all the time.

Note also the scale difference between Figs. 10(b) and 11(b). The non-model-blended peak errors are about ten times larger in the scenario with wide temperature range. There is a clear need for model blending in this case.

5.5. *Scenario 3: wide SOC range and wide temperature range*

In the final scenario, both SOC and temperature vary over a wide range. The input is ten consecutive UDDS cycles (starting at 80% SOC) and the temperature increases linearly from 0 °C to 50 °C over the 15,000 s. The results for the ROM without model blending are generated using the singe ROM linearized around 80% SOC and 0 °C. Fig. 12 shows simulation results for cell voltage. Model-blending reduces the voltage RMS error from 64.6 mV to 3.6 mV. Again, referring back to Table 3, the blended RMS error of 3.6 mV is a very reasonable value to expect when traversing the table from 0 °C and 80% SOC to 50 °C and 25% SOC.

It is interesting that the RMS error in this case is lower than when SOC is kept nearly constant but temperature has wide variation. The reason for this is that temperature changed much more quickly in Scenario 2 than it does in Scenario 3. In Scenario 2, temperature changed by 50 °C in only two UDDS cycles (3000 s). In Scenario 3, temperature changes by the same amount in ten cycles (15,000 s). The blending approach works best when the instantaneous operating point moves slowly through the space of precomputed models. However, the results of Scenario 2 show that the method still works well when the instantaneous operating point moves quite quickly.

Figs. 13–17 show the full-order results from the PDE simulation and the errors for both the model-blending and single ROM approach for all five of the internal electrochemical properties we might be interested in modeling. The ROM is able to estimate these properties at any location across the cell; for brevity, these plots show results only at the electrode/separator boundaries, because they reflect the greatest dynamic behavior and are typically the most difficult to model. In all cases, we see good modeling by the model-blended simulations, and reduced error with respect to the non-model-blended case.

6. Conclusion

We have previously shown how to create a physics-based discrete-time state-space reduced-order model of an electrochemical cell that accurately predicts both internal cell variables and the cell's voltage at a single state-of-charge and temperature setpoint. In this paper, we show how model blending can be used to create a time-varying ROM that is able to accurately track these variables over a wide range of state-of-charge and temperature. We have shown that the resulting ROM is stable, and can be implemented using commonly used design tools such as Simulink via the Simulink Coder. Simulation results demonstrate that model-blending significantly improves the accuracy of the estimated electrochemical variables to the point that they are feasible to use in a real-time system. For example, the simulation using the highly dynamic UDDS cycle over the operational SOC and temperature-range for an electric vehicle gave an RMS voltage error of only 3.6 mV versus the full-order PDE model.

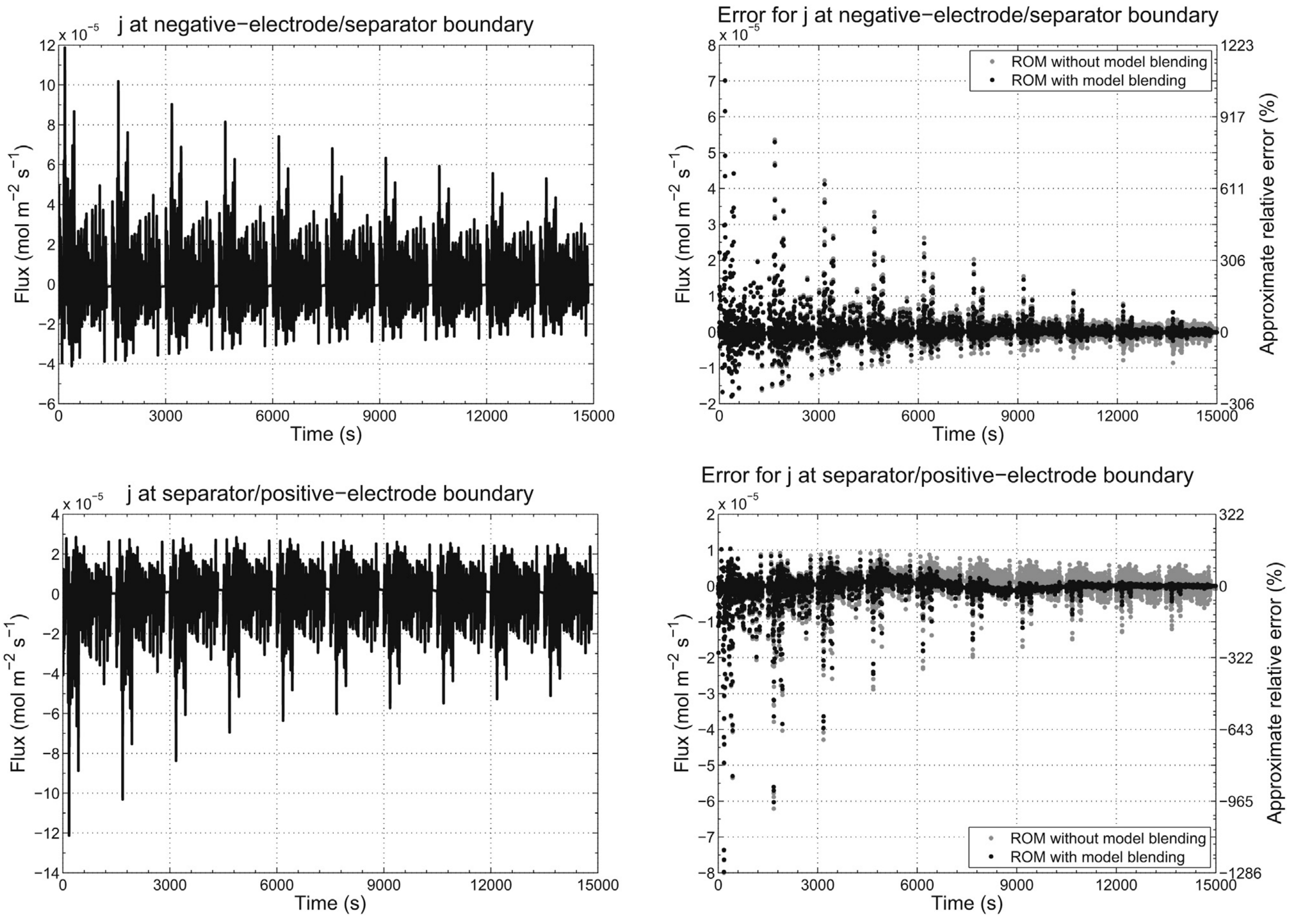


Fig. 17. *j* results for the wide SOC and temperature simulation.

Acknowledgment

Financial support for the research reported in this paper has been received from the General Motors/University of Michigan Advanced Battery Coalition for Drivetrains (GM/UM ABCD), to which the University of Colorado Colorado Springs is a subcontractor.

This work used the EAS Data Center or EAS Cloud, which is supported by the College of Engineering and Applied Science, University of Colorado Colorado Springs.

References

[1] X. Hu, S. Li, H. Peng, J. Power Sources 198 (2012) 359–367.
[2] G.L. Plett, J. Power Sources 161 (2006) 1356–1368.
[3] G.L. Plett, J. Power Sources 161 (2006) 1369–1384.
[4] G.L. Plett, IEEE Trans. Veh. Technol. 53 (2004) 1586–1593.
[5] M. Doyle, T.F. Fuller, J. Newman, J. Electrochem. Soc. 140 (1993) 1526–1533.
[6] T.F. Fuller, M. Doyle, J. Newman, J. Electrochem. Soc. 141 (1994) 1–10.
[7] K.A. Smith, Electrochemical Modeling, Estimation and Control of Lithium Ion Batteries (Ph.D. thesis), Pennsylvania State University, 2006.
[8] K.A. Smith, C.D. Rahn, C.-Y. Wang, IEEE Trans. Control Syst. Technol. 18 (2010) 654–663.
[9] K.A. Smith, C.-Y. Wang, J. Power Sources 160 (2006) 662–673.
[10] J.L. Lee, A. Chemistruck, G.L. Plett, J. Power Sources 206 (2012) 367–377.
[11] Antoulas, Sorensen, Gugercin, Contemporary Mathematics, 2001, pp. 193–219.
[12] J.L. Lee, A. Chemistruck, G.L. Plett, J. Power Sources 220 (2012) 430–448.
[13] K.A. Smith, C.D. Rahn, C.-Y. Wang, Energy Convers. Manage. 48 (2007) 2565–2578.
[14] B. Ho, R. Kalman, Regelungstechnik 14 (1966) 545–548.
[15] G.G. Botte, B.A. Johnson, R.E. White, J. Electrochem. Soc. 146 (1999) 914–923.
[16] W.B. Gu, C. Wang, in: R. Marsh, Z. Ogumi, J. Prakesh, S. Surampudi (Eds.), Lithium Batteries, Volume PV 99-25, The Electrochemical Society, 1999, pp. 748–762.
[17] D.J. Leith, W.E. Leithead, Int. J. Control 73 (2000) 1001–1025.
[18] J.S. Shamma, M. Athans, IEEE Control Syst. Mag. 12 (1992) 101–107.
[19] C.-T. Chen, Linear System Theory and Design, third ed., Oxford University Press, 1998.
[20] MATLAB, Online: http://www.mathworks.com/products/matlab/ (accessed April 2013).
[21] Simulink, Online: http://www.mathworks.com/products/simulink/ (accessed April 2013).
[22] Simulink Coder, Online: http://www.mathworks.com/products/simulink-coder/ (accessed April 2013).
[23] M. Doyle, J. Newman, A.S. Gozdz, C.N. Schmutz, J.-M. Tarascon, J. Electrochem. Soc. 143 (1996) 1890–1903.
[24] W.B. Gu, C. Wang, J. Electrochem. Soc. 147 (2000) 2910–2922.
[25] COMSOL Multiphysics, Online: http://www.comsol.com/ (accessed April 2013).
[26] E. Wood, M. Alexander, T.H. Bradley, J. Power Sources 196 (2011) 5147–5154.

Nomenclature

A: current-collector plate area of the electrode, m^2
A: state transition matrix of the state-space model
B: input matrix of the state-space model
C: output matrix of the state-space model
c: concentration of lithium in phase indicated by subscript, mol m^{-3}
$c_{e,0}$: steady-state concentration of lithium in the electrolyte phase, mol m^{-3}
$c_{s,max}$: maximum lithium concentration in an electrode particle, mol m^{-3}
$c_{s,e}$: surface concentration of lithium in a spherical electrode particle, mol m^{-3}
D: input–output coupling matrix of the state-space model
D_e^{eff}: effective electrolyte diffusivity, m^2 s^{-1}
D_s: solid diffusivity, m^2 s^{-1}
i_{app}: applied cell current, A
j: reaction flux, mol m^{-2} s^{-1}
k: rate constant for the electrochemical reaction, mol$^{\alpha-1}$ m$^{4-3\alpha}$ s^{-1}

L: length of region of cell, m
R_s: particle radius, m
t: time, s
U_{ocp}: open circuit potential, V
$\boldsymbol{x}$: state vector of state-space model
$\boldsymbol{y}$: linear output vector of state-space model

Greek

ε: volume fraction of phase indicated by subscript
Φ: potential of the phase indicated by subscript, V
κ^{eff}: effective electrolyte conductivity, S m^{-2}
σ^{eff}: effective solid conductivity, S m^{-2}

Subscript/superscript

e: pertaining to the electrolyte phase
n: pertaining to the negative electrode
p: pertaining to the positive electrode
s: pertaining to the solid phase